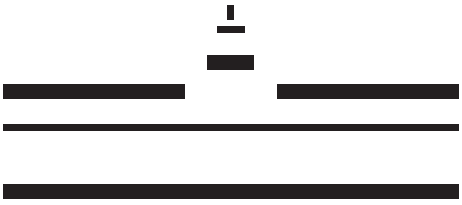


Westfälische
Wilhelms-Universität
Münster

Implementierung eines Algorithmus zur automatischen Gruppierung von Strömungslinien in zentralen Strömungen

Bachelorarbeit

vorgelegt von:
Simon Leistikow

Matrikelnummer: 406 838

Studiengang: B.Sc. Informatik

Thema gestellt von:
Prof. Dr. Klaus Hinrichs

Arbeit betreut durch:
Dipl.-Inf./-Math. Tobias Brix

Münster, 3. Oktober 2016

Inhaltsverzeichnis

1 Einleitung

In der medizinischen Forschung beschäftigt man sich mit der Analyse von Strömungen, zum Beispiel bei der Untersuchung von Krankheiten des kardiovaskulären Systems. Mittels geeigneter Aufnahmetechnik, in diesem Fall dem 4D-Phasenkontrast-MR (4D PC MRI), wird dabei ein Bereich dieses Systems gescannt, um Strömungen in diesem genauer untersuchen zu können [2].

Dabei wird meist die Trajektorie eines Teilchens in einer Flüssigkeit oder in einem Gas betrachtet. Um allerdings Strömungen ausfindig zu machen, bedarf es meist mehrerer solcher Trajektorien, die einen ähnlichen Verlauf aufweisen. Ähnlichkeit ist hierbei beispielsweise durch räumliche Nähe oder Strömungsrichtung definierbar.

In der Forschung genutzte Datensätze enthalten meist an jedem Messpunkt im dreidimensionalen Raum mindestens die lokale Strömungsrichtung und Geschwindigkeit. Daraus lassen sich Trajektorien diskretisiert durch Strömungslinien (im Folgenden auch Streamlines) berechnen, mit denen die weitere Analyse durchgeführt wird.

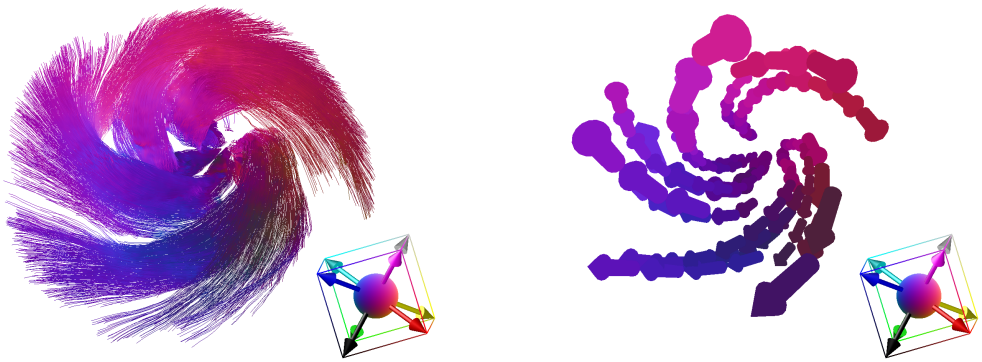
Eine rein mathematische Auswertung ist dabei meist wenig intuitiv, weshalb man sich einiger Methoden der Visualisierung bedienen kann. Eine häufig angewendete Möglichkeit besteht darin, die Strömungslinien dreidimensional zu visualisieren. Die so erhaltene Ausgabe kann dabei bei einem entsprechenden Datensatz schwierig oder gar nicht verwertbar sein, da Rauschen und die teilweise chaotische Anordnung der Strömungslinien eventuell vorhandene Strukturen verstecken.

Thema dieser Arbeit wird sein, diese Art der Visualisierung durch das automatische Detektieren von Strömungen und das Filtern von Rauschen unter Verwendung des QuickBundles-Algorithmus stark zu vereinfachen, sodass eine initiale, visuelle Analyse schnell durchgeführt werden kann. An einer solchen Auswertung ist man vor allem im medizinischen Umfeld interessiert, wenn es um eine schnelle und einfache Diagnose geht. Als Beispiel sei ein Aneurysma genannt, dessen mögliche Existenz für eine rechtzeitige Behandlung umgehend bestätigt werden muss.

Als Grundlage für die Arbeit wird das Framework Voreen (Volume Rendering Engine) dienen, welches von der Arbeitsgruppe Visualisierung und Computergrafik (VisCG) an der Universität Münster entwickelt wird [5]. Dieses ist bereits dazu in der Lage, entsprechende volumetrische Datensätze zu laden und daraus mit dem klassischen Runge-Kutta-Verfahren Strömungslinien zu berechnen und diese in einfacher Form zu visualisieren. Voreen wird bereits im medizinischen Umfeld genutzt [2].

Die Arbeit besteht also im wesentlichen aus zwei Teilen. Zum einen müssen die vom Framework berechneten Strömungslinien möglichst automatisiert gebündelt und gefiltert werden, zum anderen muss das Resultat in eine visuelle Darstellung überführt werden, beispielhaft zu sehen in Abbildung 1.1. Dafür wird eine intuitive Benutzerschnittstelle geschaffen, die eine einfache Anpassung der erhaltenen Darstellung erlaubt und sich an vorhandene Konzepte in Voreen anlehnt.

Als Anwendungsfall dient die Forschung von Philipp Rene Bovenkamp, der für die Evaluation des Algorithmus Datensätze verschiedener Komplexität zur Verfügung stellt.



(a) Vor dem Bündeln und Filtern: 5000 Strömungslinien. Der Verlauf der Strömungen ist nur schwierig zu erkennen.

(b) Nach dem Bündeln und Filtern: 16 Bündel. Der Verlauf der Strömung und deren Richtung ist klar zu erkennen.

Abbildung 1.1: Zu sehen sind zwei Darstellungsmöglichkeiten eines Phantom-Datensatzes (siehe Abschnitt 2.1), bei dem eine Schraube in den Strom der Flüssigkeit gesetzt wurde, die eine helixförmige Strömung erzeugt.

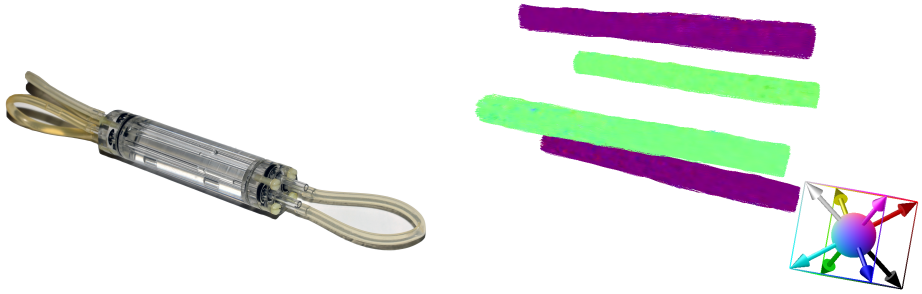
2 Grundlagen

Um die in dieser Arbeit verwendeten Algorithmen besser nachzuvollziehen, werden zunächst einige Grundlagen erklärt und diskutiert. Neben einem Überblick über die Methodik zur Erstellung der Datensätze gibt es eine Einführung in das Framework Voreen, um dessen Verarbeitung der Daten ebenfalls möglichst gut nachvollziehen zu können.

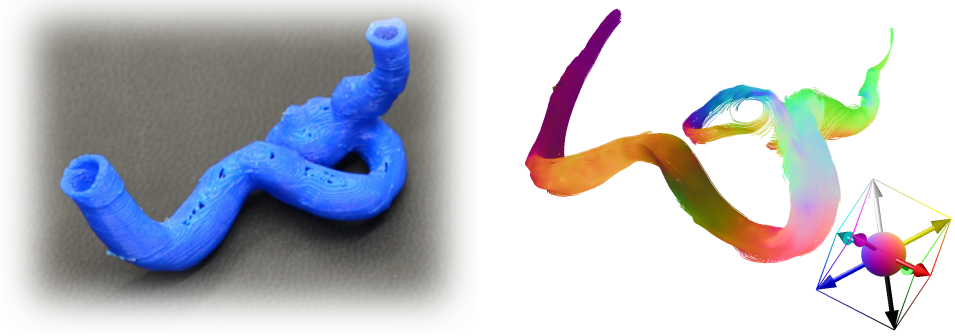
2.1 Verwendete Datensätze

Die im Rahmen dieser Arbeit betrachteten Datensätze sind volumetrisch und enthalten an jedem Messpunkt die Geschwindigkeit und Richtung der dort gemessenen Strömung. Zudem ist die Information enthalten, wie groß der Abstand zwischen je zwei Messpunkten ist (in mm). Diese Information ist für die weitere Arbeit mit dem Datensatz von Bedeutung, da sich arbiträre Einheiten für die Konfiguration der Parameter der verwendeten Algorithmen deutlich weniger gut eignen, als gebräuchliche Einheiten. Das Programm bzw. der Algorithmus soll schließlich von Benutzern bedient und konfiguriert werden, die keine Kenntnisse über den zugrundeliegenden Algorithmus besitzen, hingegen jedoch über den verwendeten Datensatz.

Die Datensätze werden mittels 4D-Phasenkontrast-MR (4D PC MRI) erstellt [2]. Dabei wird zwischen sogenannten „in vitro“ und „in vivo“-Messungen unterschieden. Ersteres bezeichnet das Erfassen von Daten an künstlich hergestellten Objekten, sogenannten Phantomen (zu sehen in Abbildung 2.1). Diese lassen sich in reine Testobjekte und Nachbildungen eines biologischen Objektes differenzieren. Ein Zylinder aus Acrylglas, wie in Abbildung 2.1(a) zu sehen, ist ein solches Testobjekt. Er kann mit verschiedenen Aufsätzen bestückt werden, die verschiedene Formen von Verwirbelungen erzeugen, wenn man eine Flüssigkeit hindurch leitet, dessen Messungen dann ausgewertet werden können. Möchte man konkreter die Strömungen innerhalb eines biologischen Objektes analysieren, ohne am lebenden Objekt messen zu müssen, so bildet man entsprechende Regionen am Computer nach. Das so erhaltene Modell wird dann mit einem 3D-Drucker erstellt und ist bereits sehr gut mit seinem Original in Bezug auf das Strömungsverhalten vergleichbar [1]. Dennoch wird der 3D-Druck erneut mithilfe von Flüssiglatex nachgebildet, um das Strömungsverhalten insbesondere an den Gefäßwänden exakter nachempfinden zu können. Eine so erstellte Nachbildung eines Aneurysmas ist in Abbildung 2.1(b) zu sehen. Solche Phantome werden genutzt, um neue Methoden



(a) Ein Zylinder aus Acrylglas, der von einer Flüssigkeit durchströmt werden kann [2]



(b) Die Nachbildung eines Aneurysmas, erstellt mittels eines Nachgusses aus Flüssiglatex eines 3D-Drucks, von Philipp Rene Bovenkamp

Abbildung 2.1: Zwei verschiedene Phantome zur Durchführung von „in vitro“-Messungen, durch die Flüssigkeit geleitet wird. Daneben jeweils die Visualisierung des durch 4D PC MRI erhaltenen Datensatzes in Voreen, mit richtungsbasierter Farbkodierung (siehe Abschnitt 2.4).

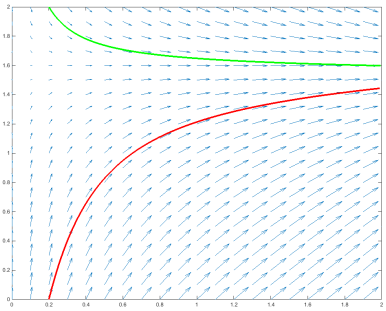
erforschen und bekannte verbessern zu können, ohne auf lebende Objekte angewiesen sein zu müssen. Letzteres ist meist deutlich aufwendiger und teurer, sowie ethisch fraglich. Daher wird eine Methode erst anhand dieser Phantome erprobt und validiert.

Führt man eine sogenannte „in vivo“-Messung am lebenden Objekt durch, beispielsweise an einer Maus, beschränkt man sich meist auf eine bestimmte Region. Im folgenden Beispiel wurde die Herzregion gewählt, zu sehen weiter unten in Abbildung 2.4. Die Maus wurde für die Zeit der etwa eineinhalbstündigen Messung narkotisiert und ihr Herzschlag dabei konstant gehalten [2]. Die bei den Messungen erhaltenen Daten werden getrennt nach räumlicher Dimension, sowie Meta-Informationen, wie etwa dem Abstand der Messpunkte, hinterlegt, sodass ein Datensatz letztlich aus mehreren Dateien besteht.

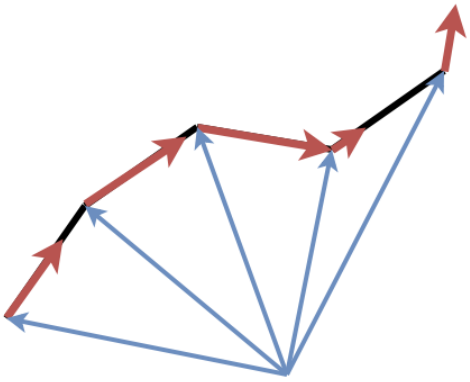
Im Folgenden wird ein solcher Datensatz jedoch mathematisch erfasst durch eine vektorielle Funktion

$$f : \mathbb{R}^3 \to \mathbb{R}^3,$$

welche jedem Punkt im Datensatz einen Richtungsvektor zuordnet, dessen Magnitude die Geschwindigkeit der Strömung am eingegebenen Datenpunkt angibt. Alle übrigen Informationen, wie der Abstand der Messpunkte, werden der Einfachheit halber als konstant über alle Datensätze angenommen.



(a) Strömungslinien im zweidimensionalen Vektorfeld



(b) eine diskretisierte Strömungslinie

Abbildung 2.2: Links sind zwei Strömungslinien (rot und grün) in einem zweidimensionalen Vektorfeld zu sehen. Die Pfeile geben die Richtung an ausgewählten Punkten im Feld an, wobei die Länge ihre Magnitude repräsentiert. Rechts ist schematisch eine diskretisierte Strömungslinie bestehend aus fünf Elementen skizziert. Die blauen Pfeile markieren die Ortsvektoren der einzelnen Elemente, die roten repräsentieren deren Richtungsvektoren. Die Länge der Richtungsvektoren muss dabei nicht der Länge der Segmente entsprechen, die je zwei Elemente miteinander verbinden. Der schwarze Pfad repräsentiert die Zusammengehörigkeit der Elemente.

2.2 Definition einer Strömungslinie

Mit Strömungslinien bezeichnet man im allgemeinen Kurven in einem Geschwindigkeitsfeld, deren Tangentenrichtung an jedem Punkt mit den Richtungen der Geschwindigkeitsvektoren im Feld übereinstimmen (zu sehen in Abbildung 2.2(a)) [7, 11]. Damit können sie eine geometrische Vereinfachung einer Strömung innerhalb eines solchen Datensatzes darstellen und repräsentieren somit eine Diskretisierung eines beliebig genauen Datensatzes. Eine weitere

Eigenschaft ist, dass sich zwei Strömungslinien in keinem Punkt schneiden können, da in einem Punkt des Geschwindigkeitsfeldes nicht zwei verschiedene Geschwindigkeiten herrschen können. Eine solche Strömungslinie sei nun definiert über eine vektorielle Funktion $s' : [0, 1] \subset \mathbb{R} \to \mathbb{R}^3$, wobei die Stelle 0 den Beginn und der Wert 1 das Ende der Strömungslinie s' im Raum definiert. In der Theorie sind diese Strömungslinien also mathematische Funktionen und weisen daher auch eine beliebige Genauigkeit auf. Im Rahmen dieser Arbeit wird diese Funktion jedoch numerisch ermittelt, was eine weitere Diskretisierung entlang der Strömungslinie zur Folge hat (siehe Kapitel 3.1).

Diese Diskretisierung ist nicht zuletzt für die Visualisierung unabdingbar, da diese nur mit endlicher Genauigkeit durchgeführt werden kann. Zudem lassen sich die Informationen, die eine solche Strömungslinie beschreiben, von ihrem Startpunkt im dazugehörigen Datensatz ausgehend, linearisiert abspeichern. Dies vereinfacht die Weiterverarbeitung der Daten und ist schlussendlich erforderlich wegen der Arbeitsweise der verwendeten Algorithmen in dieser Arbeit.

Eine solche, diskretisierte Strömungslinie s (zu sehen in Abbildung 2.2(b)) bestehend aus $k \in \mathbb{N}$ Elementen, ist im Folgenden definiert durch eine $k \times 3$ Matrix, wobei die drei Elemente jeder Zeile die Position im Raum beschreiben und die erste Zeile somit die Position des ersten Elements in Strömungsrichtung. Die Schreibweise $s = [r_1, r_2, \ldots, r_k]$ erlaubt im Folgenden eine bei 1 beginnende, indizierte Referenzierung der Elemente. Da Strömungslinien immer in Kombination eines Geschwindigkeitsfeldes berechnet werden, ist die lokale Geschwindigkeit nicht Bestandteil der Definition, sondern ist durch den Funktionswert der Funktion f (siehe vorangegangener Abschnitt) an der Position des entsprechenden Elements gegeben, d. h. $v_i = f(r_i), i = 1, \ldots, k$.

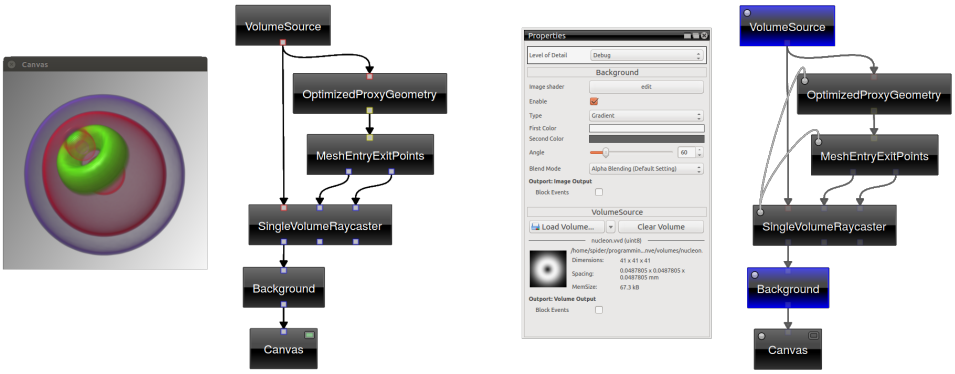
2.3 Definition eines Bündels

Ein wesentlicher Bestandteil dieser Arbeit ist das Bündeln von Strömungslinien. In der Literatur [3] ist ein solches Bündel definiert durch ein Tripel $c = (I, h, n)$, wobei im Folgenden jedes Element mit dem $\circ$-Operator referenziert werden kann. Hierbei ist I eine Liste, die alle Indizes der im Bündel enthaltenen Strömungslinien s_i enthält, n ist deren Anzahl und h deren Aufsummierung, d.h. $h = \sum_{i=1}^{k} s_i$, wobei die Summe $\sum$ hier die Matrixaddition repräsentiert. Diese kann natürlich nur durchgeführt werden, wenn die Dimensionen der Matrizen identisch sind, also die Strömungslinien dieselbe Anzahl an Elementen haben. Dieses Problem wird durch Resampling mittels linearer Interpolation gelöst, was in Kapitel 3 beschrieben wird. Die Listen I verschiedener Bündel sind disjunkt, da die Zugehörigkeit einer Strömungslinie zu einem Bündel eindeutig ist. Ein weiterer wichtiger Begriff ist der *Centroid*. Dies ist die repräsentative Strömungslinie v eines Bündels und wird ausgehend vom Tripel c berechnet durch $v = c \circ h / c \circ n$, wobei hier die übliche Matrix-Skalar-Division gemeint ist.

2.4 Das Voreen-Framework

Voreen (Volume Rendering Engine) ist ein Framework zur Visualisierung medizinischer Volumendaten, welches von der Arbeitsgruppe Visualisierung und Computergrafik (VisCG) an der Universität Münster entwickelt wird [5].

Das Framework ist modular aufgebaut, sodass dessen tatsächliche Funktionalität frei konfigurierbar ist und basiert auf der Programmiersprache C++ und der OpenGL-API [9]. Im Rahmen dieser Arbeit wird der Fokus im Wesentlichen auf die Module Flowreen, welches Algorithmen zur Fluss-Visualisierung enthält und Bovenkamp, benannt nach Philipp Rene Bovenkamp, welches das Einladen eines von ihm entwickelten Dateiformats unterstützt, gelegt.



(a) Beispielnetzwerk mit dessen Ausgabe im dazugehörigen Canvas

(b) Beispielnetzwerk im Modus zum Konfigurieren der „Property-Links“, sowie die Einstellsmöglichkeiten zweier Prozessoren (blau markiert)

Abbildung 2.3: Zu sehen ist ein simples Beispielnetzwerk in Voreen. Die linke Abbildung zeigt die Ausgabe des Bilddatenflusses, welcher über die blauen Ports übertragen und durch Übermitteln der Daten in den *Canvas*-Prozessor dargestellt wird. Die rechte Abbildung zeigt dasselbe Netzwerk mit dessen „Property-Links“. Die Blöcke sind die Prozessoren, welche ihre Daten über die Verbindungen (dunkle Pfeile) miteinander austauschen. Die hellgrauen Verbindungen sind die „Property-Links“, über welche z.B. die Ausrichtung der Kamera zwischen den Prozessoren synchronisiert werden kann.

2.4.1 Workflow und dessen Komponenten

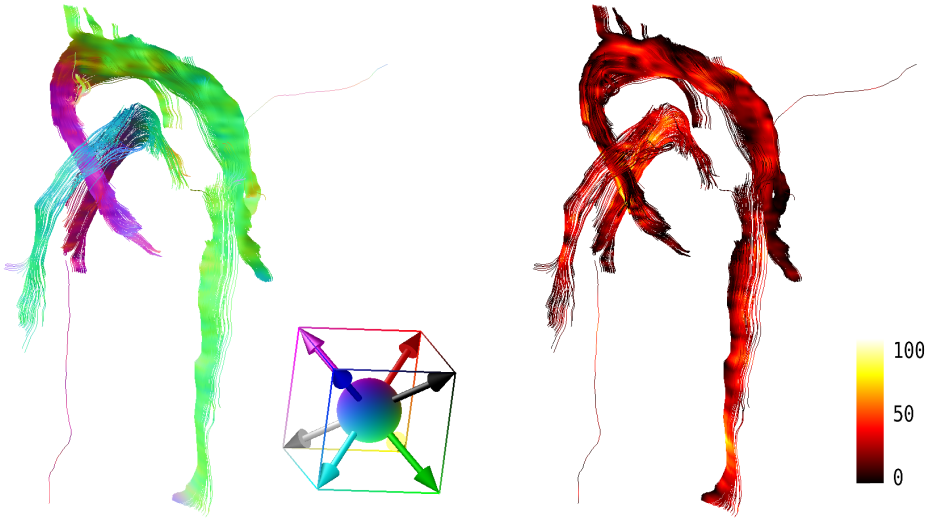
In Voreen wird der Datenfluss über ein Netzwerk abgewickelt. Dieses Netzwerk besteht aus Prozessoren, welche miteinander verbunden werden können. Jeder Prozessor stellt einen Knoten im Netzwerk dar, welcher je nach Funktionalität externe Daten einlesen, neue generieren, eingehende manipulieren oder

visualisieren kann. Jeder Prozessor besitzt mehrere Einstellungsmöglichkeiten und Ports, über welche die Prozessoren untereinander verbunden sind. Dabei sind die Einstellungsmöglichkeiten über sogenannte „Property-Links“ miteinander verbunden, damit beispielsweise die Ausrichtung der Kamera in allen Prozessoren synchronisiert ist (zu sehen in Abbildung 2.3). Über die Ports wird der Datenaustausch eingeschränkt und gesteuert. Es wird unterschieden zwischen eingehenden und ausgehenden Ports, womit die Datenflussrichtung im Netzwerk definiert wird. Dabei sind die eingehenden Ports im Allgemeinen an der Oberseite und ausgehende an der Unterseite eines Prozessors angebracht, wodurch der Datenfluss von oben nach unten verläuft. Jeder Port ist dazu in der Lage, eine fest definierte Datenstruktur entgegenzunehmen und an den Prozessor weiterzugeben, wodurch die Verbindungsmöglichkeiten zwischen den Prozessoren festgelegt werden. Dabei wird jede Datenstruktur durch eine unterschiedliche Farbe des übertragenden Ports dargestellt. Der Zustand eines Datenpakets, welches durch das Netzwerk geschickt wird, lässt sich an jedem Port einsehen, sodass der Datenfluss transparent ist. Eine besondere Rolle nimmt dabei der *Canvas*-Prozessor ein. Dieser kann das Ende des Bilddatenflusses definieren, da er keine ausgehenden Ports besitzt. Seine wesentliche Funktionalität besteht darin, die eingehenden Bilddaten anzuzeigen, weshalb er Bestandteil von nahezu jedem Netzwerk ist.

Mithilfe dieses Netzwerkkonzepts und einer gewissen Auswahl von grundsätzlich mitgelieferten Prozessoren, die unterschiedlichste Aufgaben erfüllen, lassen sich schnell Visualisierungen verschiedenster Datensätze erstellen. Voreen ist damit ein Rapid-Prototyping-Framework [5]. Die Funktionalität ist dabei durch viele verfügbare Module frei erweiterbar. Jedes dieser Module fügt neue Prozessoren hinzu, die mit den vorhandenen verbunden werden können und so das Netzwerk erweitern. Des Weiteren werden alle Informationen über ein Netzwerk und die Konfiguration der einzelnen Prozessoren in einem Workspace gespeichert. Diese dienen der Persistierung des Zustandes der Anwendung mittels der Speicherung in einem, auf XML basierten und für den Menschen lesbaren VWS-File (Voreen Workspace). Nach dem Wiedereinladen in Voreen ist das Weiterarbeiten an genau dem Punkt der Speicherung möglich, beispielsweise mit den gleichen Fenster- und Kamerapositionen. Auf diese Weise können zudem verschiedene Applikationen, wie im folgenden Abschnitt beschrieben, bereitgestellt und weiterentwickelt werden.

2.4.2 Das Flowreen Modul

Flowreen ist ein Modul von Voreen, welches Funktionalitäten zur Visualisierung von Strömungen bietet. Es liefert dafür eigene Datenstrukturen und Prozessoren, die im Rahmen dieser Arbeit verwendet und erweitert werden. Die Kernfunktionalität besteht dabei in der Erzeugung von Strömungslinien aus einem volumetrischen Datensatz (siehe Abschnitt 2.1) sowie deren Visualisierung und Export. Die Prozessoren, die diese Arbeit übernehmen, sind der *Streamli-*



(a) Richtungsbasierte Farbkodierung

(b) Geschwindigkeitsbasierte Farbkodierung (Werte geben die Geschwindigkeit in % relativ zum höchsten Messwert im Datensatz an)

Abbildung 2.4: Die verschiedenen Möglichkeiten der farblichen Visualisierung von Strömungslinien, verdeutlicht anhand des Herzens einer Maus.

neCreator, *StreamlineRenderer3D* und *StreamlineSave*. Die Prozessoren lassen sich direkt miteinander verbinden, sodass die erstellten Strömungslinien ohne Umwege visualisiert und exportiert werden können.

Weiterhin existiert ein Prozessor zum Filtern der erstellten Datensätze, den *StreamlineSelector*. Dieser erlaubt anhand mehrerer Einstellungsmöglichkeiten eine Auswahl der eingehenden Strömungslinien vorzunehmen, wie etwa durch das Entfernen derer, die sich nicht innerhalb einer selbstdefinierten Box (Region Of Interest) befinden. Unabhängig von dieser Filterung lassen sich Strömungslinien mittels des *StreamlineRenderer3D*-Prozessors in Form von farbkodierten Linien visualisieren. Es kann dabei zwischen einer richtungsbasierten (Abbildung 2.4(a)) und einer geschwindigkeitsbasierten (Abbildung 2.4(b)) Farbkodierung gewählt werden. Die Interpretation des Farbwertes kann dabei anhand des entsprechenden Overlays abgelesen werden, zu sehen jeweils rechts unten in den Abbildungen. Der Export kann sowohl in ein Voreen-eigenes Dateiformat, als auch in eine CSV-Datei (Comma Separated Value) erfolgen.

2.4.3 Das Bovenkamp Modul

Für Philipp Rene Bovenkamp, dessen Forschung auf dem Gebiet der Strömungsanalyse im medizinischen Umfeld diese Arbeit unterstützen soll, wurde von der Arbeitsgruppe Visualisierung und Computergrafik (VisCG) ein ei-

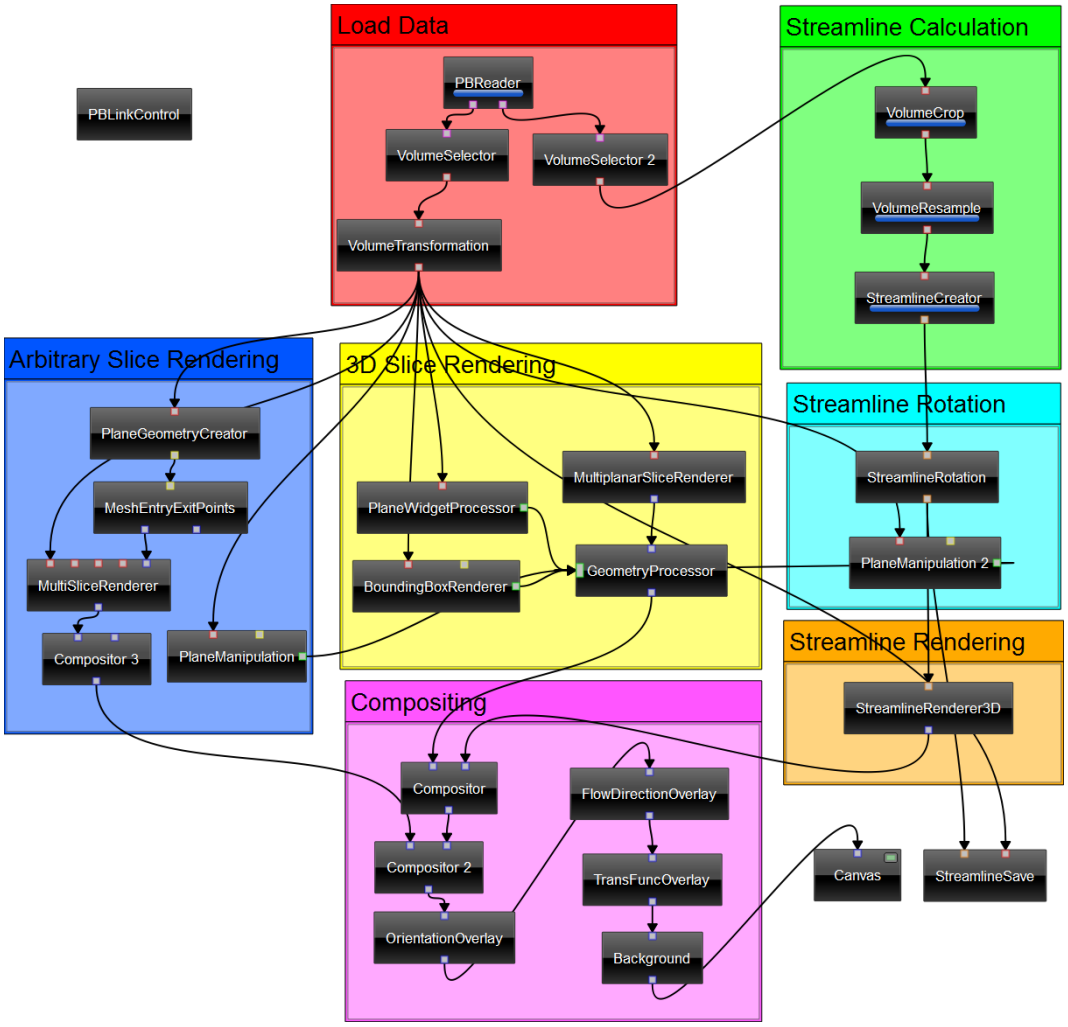
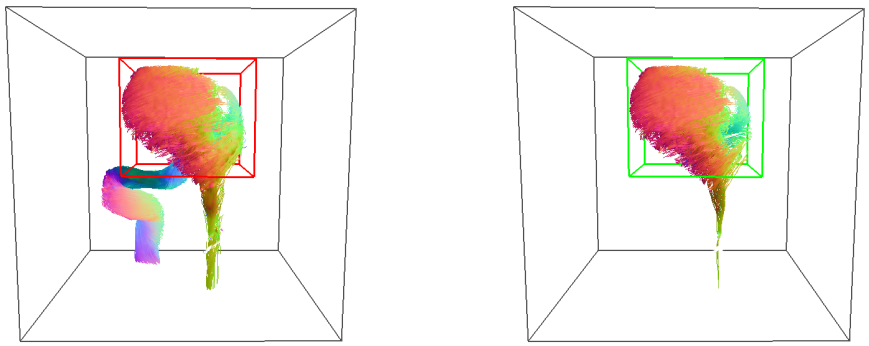


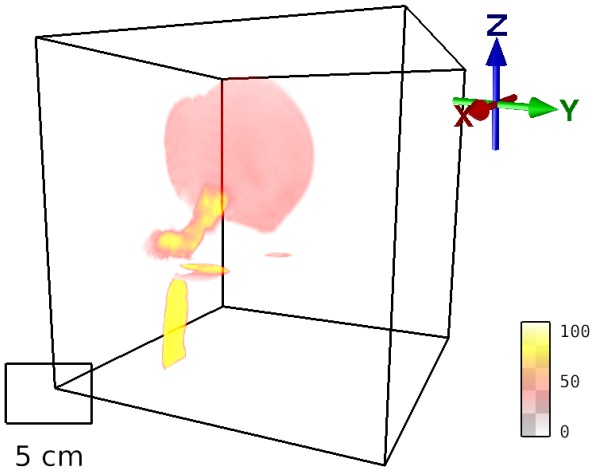
Abbildung 2.5: Netzwerk des *Create*-Workspaces

genes Modul entwickelt. Dieses hat die Aufgabe, die von ihm mit externer Software erstellten Datensätze einlesen und bearbeiten zu können. Für diesen Zweck enthält es den *Create*-, *Select*- und den *Render*-Workspace, welche drei verschiedene Anwendungsbereiche abdecken. Ersterer beinhaltet die Funktionalität zum Einladen der genannten Datensätze und zum Erstellen der Strömungslinien, welche zugleich visualisiert und exportiert werden können. Der *Select*-Workspace erlaubt das Nachbearbeiten eines so erstellten Datensatzes durch die Selektion von Strömungslinien in bestimmten Bereichen, wobei der *Render*-Workspace hingegen lediglich die Möglichkeit bietet, einen beliebigen Strömungsliniendatensatz beispielsweise für Demonstrationszwecke zu visualisieren.

Der *Create*-Workspace (siehe Abbildung 2.5) enthält im Wesentlichen die Funktionalität der beiden anderen Workspaces, weshalb er im Folgenden repräsentativ genauer betrachtet wird. Er bietet zunächst die Möglichkeit, einen



(a) Eine ausgewählte Region innerhalb eines Datensatzes, sowie die enthaltenen Strömungslinien vor (links) und nach (rechts) der Selektion



(b) drei orthogonale Schnitte durch einen Datensatz

Abbildung 2.6: Verschiedene Funktionalitäten, bereitgestellt durch das Bovenkamp Modul.

entsprechenden Volumendatensatz (siehe Abschnitt 2.1) einzuladen (markiert in rot). Bei den beiden anderen Workspaces wird an dieser Stelle hingegen ein Strömungsliniendatensatz eingeladen, dessen Herkunft im Folgenden erklärt wird. Die Daten werden nun an mehreren Stellen für unterschiedliche Zwecke ausgewertet. Das Erstellen von Strömungslinien aus dem Datensatz (markiert in grün) ist dabei die diesen Workspace auszeichnende Funktionalität und ist in den beiden anderen nicht enthalten. Im *Select*-Workspace wird dagegen durch sogenannte *StreamlineSelector*-Prozessoren das Selektieren von Strömungslinien ermöglicht, wie im vorangegangenen Abschnitt erklärt (siehe Abbildung 2.6(a)). Die erstellten Strömungslinien können daraufhin optional im Raum rotiert (markiert in hellblau) und visualisiert werden (markiert in orange). Letzteres wird dabei unterstützt durch die Darstellung von sogenannten Schnitten durch den Datensatz, sowie von diversen Overlays zum Ermit-

teln seiner Ausrichtung, Größe und Farbkodierung (siehe Abbildung 2.6(b)). Zum einen gibt es die Möglichkeit einen beliebigen, vom Benutzer festgelegten Schnitt durch den Datensatz darzustellen (markiert in dunkelblau), zum anderen können drei paarweise orthogonale Schnitte gerendert werden, welche die Orientierung im Datensatz erleichtern sollen (markiert in gelb). All diese Darstellungen werden letztlich zu einem Bild zusammengefügt (markiert in lila) und können durch einen *Canvas*-Prozessor ausgegeben werden. Zusammenfassend lässt sich die Idee hinter den Workspaces mit folgendem Workflow festhalten. Ein Volumendatensatz wird mit dem *Create*-Workspace eingeladen und daraus mit den beschriebenen Visualisierungsmöglichkeiten ein Strömungsliniendatensatz erstellt. Dieser wird exportiert und im *Select*-Workspace importiert, wo enthaltene Strömungslinien weiter aussortiert werden. Der so aufbereitete Datensatz wird erneut exportiert und im *Render*-Workspace importiert. Dort werden die Strömungslinien zusammen mit drei orthogonalen Schnitten, beispielsweise zu Demonstrationszwecken, visualisiert.

3 Algorithmen

In diesem Kapitel wird das automatisierte Gruppieren von Strömungslinien algorithmisch erfasst. Dafür wird kurz die Funktionsweise des vorgelagerten Runge-Kutta-Verfahrens erläutert, welches die Eingabedaten des QuickBundles-Algorithmus liefert. Im Anschluss dieses Kapitels wird dann auf die Implementierung und Integration dieser Algorithmen in Voreen eingegangen.

3.1 Finden von Strömungslinien (Runge-Kutta)

Das Finden von Strömungslinien erfolgt unter Berücksichtigung mehrerer dynamisch festlegbarer Parameter mittels des klassischen Runge-Kutta-Verfahrens [8], welches im Modul Flowreen (siehe Abschnitt 2.4.2) implementiert ist.

Dafür werden unter Verwendung eines Zufallszahlengenerators innerhalb des Volumendatensatzes Positionen bestimmt, die als Startpunkte für die Berechnung der Strömungslinien verwendet werden.

Sei $f : \mathbb{R}^3 \to \mathbb{R}^3$ die Funktion, welche zu jedem Punkt des Datensatzes die lokale Geschwindigkeit liefert (siehe Abschnitt 2.1), dann berechnet sich ein Element r_i der Strömungslinie, ausgehend vom Startpunkt r_0 unter Verwendung der Schrittweite h durch

$$r_{i+1} = r_i + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6}$$

mit

$$\begin{aligned} k_1 &= \hat{f}(r_i) \cdot h, \\ k_2 &= \hat{f}(r_i + \tfrac{k_1}{2}) \cdot h, \\ k_3 &= \hat{f}(r_i + \tfrac{k_2}{2}) \cdot h, \\ k_4 &= \hat{f}(r_i + k_3) \cdot h, \end{aligned}$$

wobei $\hat{f}$ die benötigte Normalisierung von f denotiert.

Von den zufällig bestimmten Startpunkten ausgehend wird nachfolgend numerisch mit dem genannten Verfahren in und entgegen der lokalen Strömungsrichtung der Datensatz traversiert, bis ein Abbruchkriterium eintritt. Dies ist der Fall, wenn die Grenzen des Datensatzes erreicht wurden oder die Geschwindigkeit der lokalen Strömung den eingestellten Schwellwert über- oder unterschreitet.

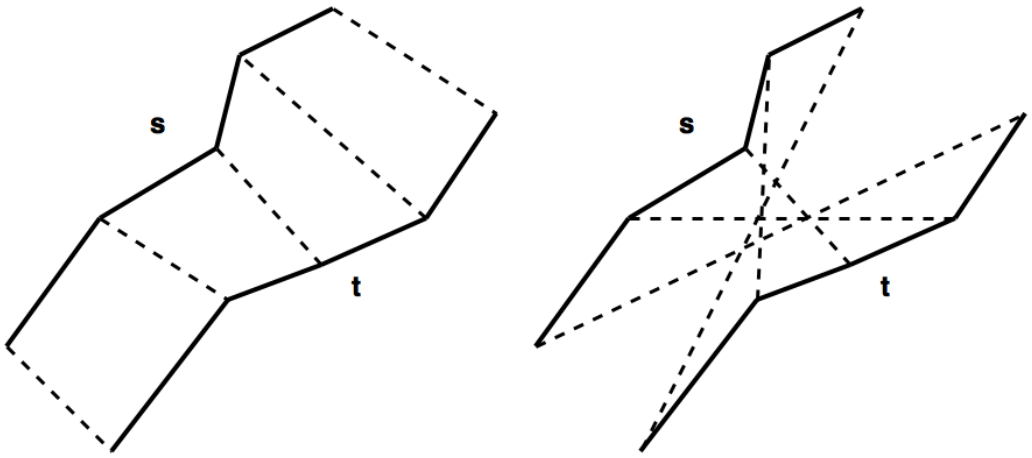
Algorithm 1 Modifizierter QuickBundles-Algorithmus [3]

Input: $S = [s_1, \ldots, s_i, \ldots, s_n], \varepsilon$ // Liste von Strömungslinien und Schwellwert
Output: $C = [c_1, \ldots, c_i, \ldots, c_m]$ // Liste von Bündeln

$C \leftarrow$ **empty** // Zu Beginn gibt es keine Bündel
for $i = 1, \ldots, n$ **do**
    $t \leftarrow resampled(s_i)$ // Resampling der Strömungslinie
    $d_{min} \leftarrow \infty$
    $l \leftarrow 0$
    $m \leftarrow$ **length**(C)
    for $j = 1, \ldots, m$ **do**
        $v \leftarrow c_j \circ h / c_j \circ n$ // Berechnung des *Centroids* des $j$ten Bündels
        $d \leftarrow d_{MDF}(v, t)$ // Berechne die MDF-Distanz zwischen v und t
        if $d < d_{min}$ **then**
            $d_{min} \leftarrow d$ // Aktualisiere minimale Distanz
            $l \leftarrow j$ // Aktualisiere dazugehörigen Index
        end if
    end for
    if $d_{min} < \varepsilon$ **then**
        $c_l \circ h \leftarrow c_l \circ h + t$ // Aktualisiere die Summe der Strömungslinien
        $c_l \circ n \leftarrow c_l \circ n + 1$ // Aktualisiere die Anzahl der Strömungslinien
        append$(C_l \circ I, i)$ // Füge den Index in die Liste ein
    else
        $c_{m+1} \leftarrow ([i], t, l)$ // Erstelle ein neues Bündel
        append(C, c_{m+1}) // Nimm es in die Liste aller Bündel auf
    end if
end for
return C

3.2 Gruppieren von Strömungslinien (QuickBundles)

Das Finden der Bündel wurde mittels einer modifizierten Version des QuickBundle Algorithmus realisiert (siehe Algorithmus 1). Die Wahl fiel auf diesen, weil er im Vergleich zu anderen Gruppierungsalgorithmen wie *k-means* [4] auf Geschwindigkeit und Speicherbedarf optimiert ist. Er kann zudem gut mit sehr vielen Strömungslinien umgehen, da jede genau einmal betrachtet wird. Der Algorithmus iteriert dabei über alle Strömungslinien und versucht ein Bündel zu finden, dem er sie zuordnen kann. Das Entscheidungskriterium dafür, ob die Zuordnung stattfinden kann, ist durch eine Metrik d_{MDF} und einen Schwellwert ε definiert (siehe folgender Unterabschnitt), wobei bei meh-



(a) direct Distanzen (b) direct-flipped Distanzen

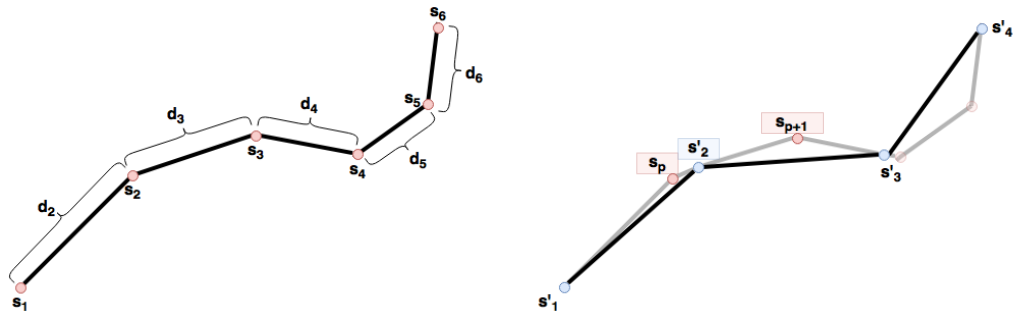
Abbildung 3.1: Vergleich der verwendeten Metriken der MDF-Distanz.

reren möglichen Bündeln jenes mit geringstem Abstand gewählt wird (Zeilen 10-13). Bei erfolgter Zuordnung wird das entsprechende Bündel um die Information ergänzt (Zeilen 16-18) und die nächste Strömungslinie wird betrachtet. Sollte kein solches Bündel gefunden werden, wird ein neues angelegt, dessen *Centroid* gerade der betrachteten Strömungslinie entspricht (Zeilen 20-21). Für das Anwenden der Metrik ist jedoch zunächst das Resampling (Zeile 3) aller Strömungslinien auf dieselbe Anzahl von Elementen erforderlich (siehe Abschnitt 3.2.2). Der Algorithmus hat keine Aktualisierungsphase der Bündel: wurde eine Strömungslinie einmal einem Bündel zugeordnet, bleibt die Zuordnung bestehen. Durch diesen Ansatz begründet sich seine Geschwindigkeit [3].

3.2.1 Abstand zwischen Strömungslinien (MDF)

Der QuickBundles - Algorithmus benötigt eine Funktion, die eine Distanz zwischen zwei Strömungslinien berechnet. Diese Distanzfunktion ist in der Literatur gegeben durch die „Minimum average direct-flip"-Distanz [3]. Diese Distanzfunktion berücksichtigt, dass die einzelnen Elemente einer diskretisierten Strömungslinien $s = [r_1, r_2, \ldots, r_k]$ nicht in Richtung der Strömung gespeichert sein müssen, sondern auch in umgekehrter Reihenfolge, notiert als $s^F = [r_k, r_{k-1}, \ldots, r_1]$. Die folgende Metrik berechnet den durchschnittlichen Abstand von zwei Strömungslinien unter paarweiser Betrachtung von je einem Punkt mit gleichem Index (siehe Abbildung 3.1(a)). Hierbei seien s, t Strömungslinien mit je $k \in \mathbb{N}$ Elementen und $|x - y|$ gebe die euklidische Distanz zwischen zwei Punkten $x, y \in \mathbb{R}^3$ an.

$$d_{direct}(s,t) = \frac{1}{k} \sum_{i=0}^{k} |r_i^{(t)} - r_i^{(s)}|$$



(a) Strömungslinie vor dem Resampling, bestehend aus sechs Elementen

(b) Strömungslinie nach dem Resampling auf vier Elemente

Abbildung 3.2: Skizze des Resampling-Prozesses mit den betrachteten Elemente und Abständen.

Die nachfolgende Metrik berechnet dieselbe Distanz wie die erste, jedoch ist eine der beiden Strömungslinien gedreht, also die Anordnung der Elemente vertauscht (siehe Abbildung 3.1(b)).

$$d_{flipped}(s,t) = d_{direct}(s,t^F) = d_{direct}(s^F,t)$$

Die MDF-Metrik berechnet das Minimum der beiden vorangegangen.

$$d_{MDF}(s,t) = min(d_{direct}(s,t), d_{flipped}(s,t))$$

Da der verwendete Runge-Kutta-Algorithmus die Anordnung der Elemente in Strömungsrichtung garantiert, vereinfacht sich die Funktion zu:

$$d_{MDF}(s,t) = d_{direct}(s,t)$$

Ferner kann diese Vereinfachung vorgenommen werden, da so Strömungslinien, die in entgegengesetzte Richtung verlaufen, erst mit einem deutlich höheren Schwellwert zusammengefasst werden, als wenn man die $d_{flipped}(s,t)$ Funktion verwenden würde. Somit ist die benötigte Distanzfunktion durch die d_{MDF}-Metrik definiert. Diese lässt sich allerdings nur auf zwei Strömungslinien mit gleicher Anzahl an Elementen anwenden, wodurch das, im folgenden Unterabschnitt erläuterte, Resampling notwendig wird.

3.2.2 Resampling

Der QuickBundles Algorithmus arbeitet mit *Centroids* (siehe Abschnitt 2.3). Für deren Berechnung ist es erforderlich, dass alle im Bündel enthaltenen Strömungslinien dieselbe Anzahl an Elementen besitzen. Zusätzlich ist es notwendig, dass die Abstände zwischen den Elementen je Strömungslinie identisch sind, damit es keine Gewichtung zwischen den Elementen des *Centroid* gibt.

Aus diesem Grund muss jede Strömungslinie vor der Betrachtung geresampled werden. Dabei werden Start- und Endpunkt beibehalten und die übrigen Punkte per linearer Interpolation bestimmt. Gegeben sei eine Strömungslinie $s = [r_1, r_2, \ldots, r_n]$ mit $n \in \mathbb{N}$ Elementen. Diese soll auf $k \in \mathbb{N}^{\geq 2}$ Elemente geresampled werden, also $s' = [r'_1, r'_2, \ldots, r'_k]$.

Zunächst wird der Abstand d_i eines jeden Elements $r_i, i = 1, 2, \ldots, n$ zum Startpunkt, also r_1 bestimmt, wobei mit Abstand die Summe der vorangegangenen Segmentlängen gemeint ist:

$$d_i = \sum_{l=1}^{i} |r_{l+1} - r_l|.$$

Dabei wird die Gesamtlänge l_{ges} von s zusätzlich bestimmt, sowie die neue, für alle Segmente identische, Länge l_{seg}:

$$l_{ges} = d_n,$$
$$l_{seg} = \frac{l_{ges}}{n-1}.$$

Nun wird für jedes neue Element r'_j die Distanz d'_j vom Startpunkt mit

$$d'_j = j \cdot l_{seg}$$

berechnet, woraufhin der Index p zum Element r'_j bestimmt wird, für den

$$d_p \leq d'_j < d_{p+1}$$

gilt. Dies lässt sich anhand von Abbildung 3.2 nachvollziehen. Jetzt kann der Koeffizient t für die lineare Interpolation mit

$$t = \frac{d'_j - d_p}{d_{p+1} - d_p}$$

berechnet werden. Schlussendlich ergibt sich nun die Position eines Elements r'_j mit linearer Interpolation:

$$r'_j = r_p \cdot (1 - t) + r_{p+1} \cdot t.$$

3.2.3 Laufzeitanalyse

Für die Ermittlung der Laufzeitkomplexität werden zunächst die ins Gewicht fallenden Parameter bestimmt. Zweifelsohne lässt sich die Anzahl der Strömungslinien n als ein solcher festhalten, da große Datensätze je nach Genauigkeit beliebig viele enthalten können. Die Anzahl der Bündel m ist für die Zuordnung einer Strömungslinie entscheidend und kann je nach Parametrisierung durch ϵ ebenfalls sehr groß werden, jedoch nicht größer als n. Der Einfluss von ϵ wird daher implizit durch m ausgedrückt. Die maximale Anzahl l von Elemente einer Strömungslinie vor dem Resampling ist ebenfalls stark von der Parametrisierung und dem Datensatz abhängig. Die Anzahl von Elementen k nach dem

Resampling kann im Allgemeinen sehr klein gewählt werden (Größenordnung 10^1) [3] und ist somit meistens kleiner, höchstens jedoch gleich l.

Die Laufzeit des Resamplings wird nun der Einfachheit halber gesondert berechnet, da sie für alle betrachteten Fälle identisch ist. Sie ist gegeben durch

$$\mathcal{O}(n \cdot (l + k)) = \mathcal{O}(n \cdot l),$$

da je Strömungslinie die Distanzen zwischen allen aufeinanderfolgenden Elementen, sowie die neuen Elemente berechnet werden müssen.

Betrachten wir nun die Laufzeit eines vollständigen Durchlaufs des QuickBundles Algorithmus im bestmöglichen Fall. Dieser tritt ein, wenn die Anzahl der Bündel minimal bleibt, also nach der Betrachtung der ersten Strömungslinie immer genau ein Bündel existiert. Damit ist die Laufzeit maßgeblich durch die Anzahl der Strömungslinien bestimmt, was zu einer Laufzeit von

$$\mathcal{O}(n) + \mathcal{O}(n \cdot l) = \mathcal{O}(n \cdot l)$$

führt.

Im durchschnittlichen Fall findet sich deutlich mehr als ein Bündel, wodurch sich die Laufzeit zu

$$\mathcal{O}(n \cdot m) + \mathcal{O}(n \cdot l) = \mathcal{O}(n \cdot (m + l))$$

verschlechtert. Da im Allgemeinen die Anzahl der Bündel jedoch deutlich kleiner ist, als die der Strömungslinien, d. h. $m \ll n$, kann die Laufzeit auch im durchschnittlichen Fall durch

$$\mathcal{O}(n) + \mathcal{O}(n \cdot l) = \mathcal{O}(n \cdot l)$$

angegeben werden [3].

Im schlechtesten Fall ist die Laufzeitkomplexität jedoch durch

$$\mathcal{O}(n^2) + \mathcal{O}(n \cdot l) = \mathcal{O}(n^2 + n \cdot l)$$

anzugeben. Für $\varepsilon = 0$ degeneriert die Anzahl der Bündel zu $m = n$. Zu diesem Resultat kommt es, da Strömungslinien sich nicht schneiden können (siehe Abschnitt 2.2) und somit der durchschnittliche Abstand niemals den Wert 0 annimmt. Dies hat letztlich zur Folge, dass es für jede Strömungslinie genau ein Bündel gibt, woraus durch die zwei verschachtelten Schleifen die quadratische Laufzeit resultiert.

3.3 Anwendbarkeit von QuickBundles

Der QuickBundles-Algorithmus bietet ein gutes Verhältnis zwischen der Qualität des Ergebnisses der Gruppierung und der dafür aufgewendeten Zeit, sowie

dem benötigten Speicherplatz. [3]. Dafür müssen allerdings gewisse Voraussetzungen erfüllt werden.

Zum einen müssen vorhandene Strömungen durch mehrere Strömungslinien repräsentiert werden, damit die Anwendung des Algorithmus überhaupt sinnvoll ist. Ansonsten mangelt es an Referenz-Strömungslinien und er wird, je nach der Wahl von ε, vermehrt auch solche zusammenfassen, die eigentlich nicht zusammengehören. Außerdem sollte die Länge der Strömungslinien möglichst maximal sein. Ist dies nicht der Fall, und eine Strömung besteht aus ähnlichen Strömungslinien, die allerdings nur eine sehr kurze Distanz über parallel verlaufen, so werden diese höchstwahrscheinlich nicht zusammengefasst. Ist der Schwellwert dagegen groß genug, würde dies zu einer starken Verkürzung an beiden Enden der erkannten Strömung führen, zu sehen in Abbildung 3.3. Die Verkürzungen Δx_1 und Δx_2 werden signifikanter, je größer der Schwellwert ist. Verringert man diesen hingegen, erhöht sich die Anzahl der Bündel.

Bei einem ungeeigneten Datensatz von Strömungslinien gibt es somit einen Trade-Off zwischen der Minimierung der Anzahl der Bündel und der Exaktheit der räumlichen Abdeckung einer Strömung. Für ein optimales Ergebnis ist es daher notwendig, den Eingabedatensatz für den Algorithmus ebenfalls möglichst optimal zu wählen.

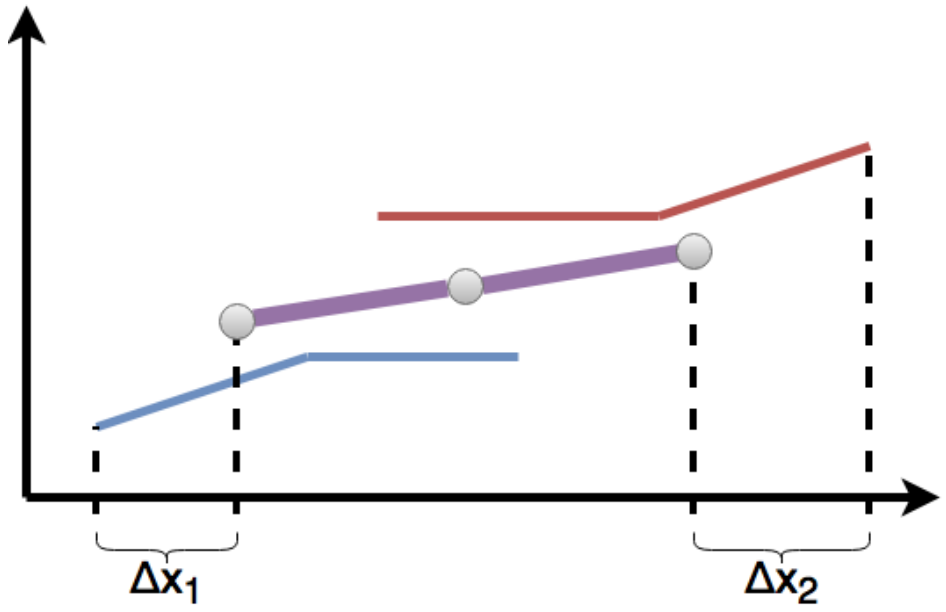


Abbildung 3.3: Skizze zweier Strömungslinien (rot und blau), die zu einem Bündel zusammengefasst werden (violett). Hier wird die Verkürzung der repräsentierten Strömung an beiden Enden sichtbar ($\Delta x_1, \Delta x_2$), die durch einen zu hoch eingestellten Schwellwert bei einem ungeeigneten Datensatz entstehen kann.

4 Implementierung von QuickBundles

Das zentrale Ziel dieser Arbeit ist die Implementierung des in Abschnitt 3.2 vorgestellten QuickBundles-Algorithmus. Dieser soll in die bestehende Implementierung von Voreen eingearbeitet werden. Dafür ist es notwendig, bestehende Datenstrukturen und Prozessoren anzupassen, sowie neue zu ergänzen. Ein besonderes Augenmerk wird dabei auf die Konsistenz zum vorhandenen Quelltext sowie auf einen möglichst modularen Aufbau gelegt. In den folgenden Abschnitten werden die einzelnen Bestandteile der Implementierung im Detail erklärt.

Die verwendete Programmiersprache ist C++. Dies ist im Wesentlichen durch die Nutzung der Sprache in dem zu erweiternden Framework Voreen begründet.

4.1 Integration in das Flowreen Modul

Der vorgestellte Algorithmus (siehe Abschnitt 3.2) wurde in das Modul Flowreen (siehe Abschnitt 2.4.2) integriert. Insbesondere wurde auf eine inkrementelle Implementierung Wert gelegt, damit die bestehende Funktionalitäten möglichst unangetastet bleiben. Dieser Abschnitt beleuchtet die im Rahmen der Implementierung notwendig gewordenen Änderungen am vorhandenen Quelltext, sowie dessen Ergänzungen.

4.1.1 Der Datenfluss

Der wesentliche Teil des Datenflusses vom Einladen eines Volumendatensatzes (siehe Abschnitt 2.1) bis hin zum Visualisieren der Strömungslinien und Bündel wird nun anhand des Klassendiagramms (siehe Abbildung 4.1) erläutert.

Nachdem ein Datensatz eingeladen wurde, startet der *StreamlineCreator*-Prozessor einen *Thread*, hier *StreamlineCreatorBackgroundThread*, der im Hintergrund der laufenden Anwendung mit dem Runge-Kutta-Verfahren Strömungslinien (*Streamline*) berechnet. Diese werden in einer Datenstruktur (*StreamlineList*) gespeichert, die vom *StreamlineCreator*-Prozessor verwaltet wird. Die *Streamlines* werden nach Abschluss der Berechnung bereits über einen entsprechenden Port (siehe Abschnitt 2.4) freigegeben, sodass ein angebundener *StreamlineRenderer3D*-Prozessor diese visualisieren kann. Im Hintergrund

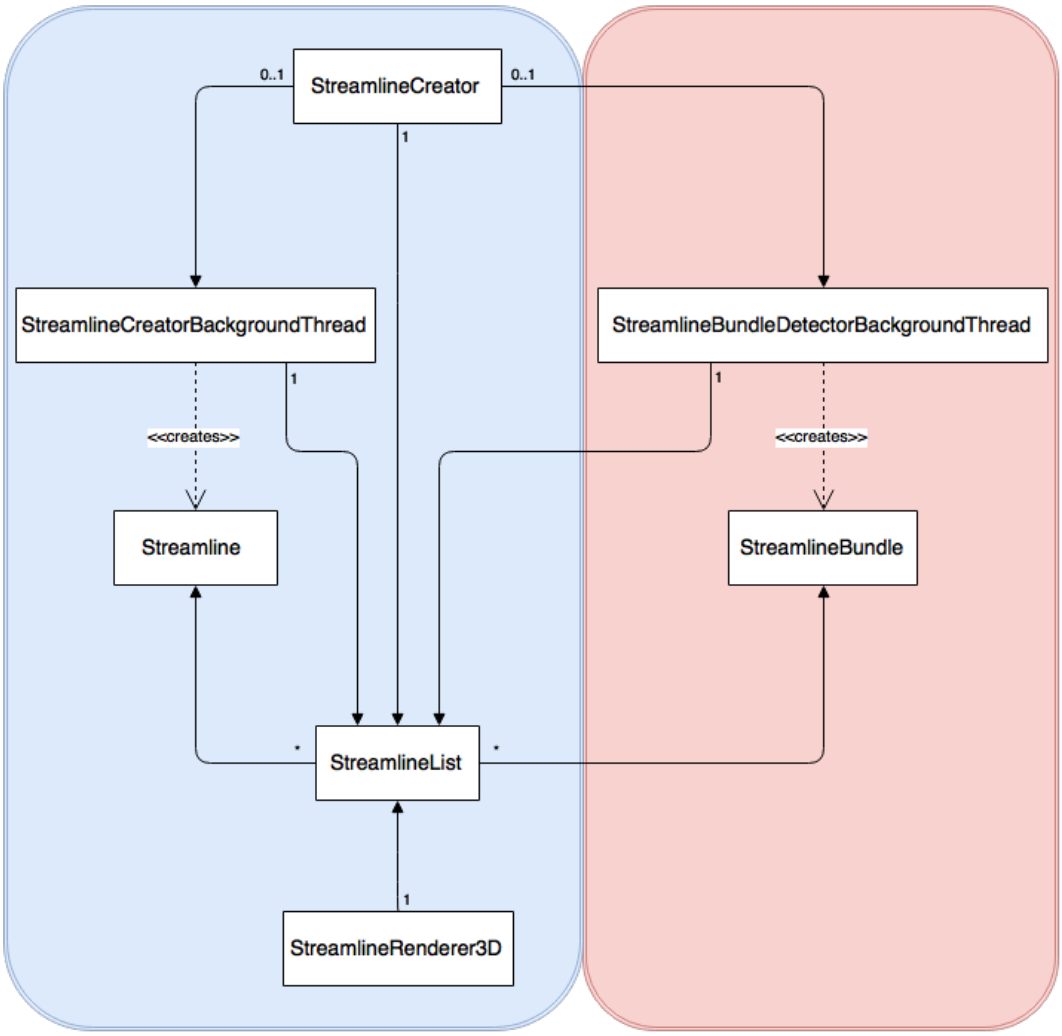


Abbildung 4.1: UML-Diagramm der wichtigsten Klassen. Der blaue Bereich beinhaltet die veränderten, der rote die hinzugefügten Klassen.

wird in der Zwischenzeit vom *StreamlineCreator*-Prozessor ein neuer Hintergrundthread gestartet, hier *StreamlineBundleDetectorBackgroundThread*. Dieser führt den QuickBundles-Algorithmus auf den zuvor berechneten *Streamlines* aus und speichert die gefundenen Bündel (*StreamlineBundle*) in derselben *StreamlineList*-Datenstruktur, die auch die *Streamlines* speichert. An dieser Stelle wird deutlich, warum eine Verwendung von Hintergrundthreads sinnvoll ist. So lässt sich die Visualisierung der Strömungslinien bereits uneingeschränkt betrachten, während die Berechnung der Bündel abläuft. Zudem können so die Parameter auch jederzeit angepasst werden. Sollte die Berechnung noch nicht abgeschlossen sein, wird sie abgebrochen und startet unmittelbar erneut, mit der neuen Konfiguration.

Der QuickBundles-Algorithmus wurde noch um eine recht simple Rauschunterdrückung erweitert. Dafür wird ein weiterer Schwellwert definiert, der

die Mindestanzahl an Strömungslinien innerhalb eines Bündels angibt. Nach dem Durchlaufen des Algorithmus wird jedes Bündel erneut betrachtet und die darin enthaltene Anzahl mit dem Schwellwert verglichen. Wird dieser unterschritten, so wird das Bündel verworfen und die enthaltenen Strömungslinien als Rauschen klassifiziert. Dafür wurde die *StreamlineList*-Datenstruktur erweitert, sodass sie zu den enthaltenen *Streamlines* noch die gefundenen *StreamlineBundles*, sowie eine Liste speichert, welche die als Rauschen klassifizierten *Streamlines* beinhaltet.

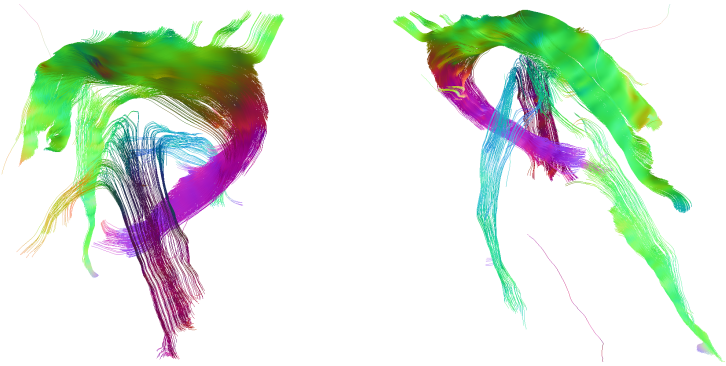
Dieser simple Ansatz ist effektiv, da bereits ein Schwellwert von einem Prozent der Gesamtzahl der Strömungslinien das meiste Rauschen filtern kann, was experimentell ermittelt wurde. Dennoch sollte der Algorithmus ohnehin auf einem Datensatz operieren, der möglichst frei von Rauschen ist.

4.1.2 Formen der Visualisierung

Neben dem Gruppieren der Strömungslinien ist die Überführung der Bündel in eine möglichst intuitive, visuelle Darstellung eine entscheidende Aufgabe. Dafür wurden für unterschiedliche Anforderungen verschiedene Darstellungsmöglichkeiten in den *StreamlineRenderer3D*-Prozessor integriert, welche die vorhandene Strömungslinienvisualisierung ergänzen (siehe Abbildung 4.2(a)). Der Benutzer kann zwischen einer Liniendarstellung, einer Zylinderdarstellung und einer Pfeildarstellung wählen. Jede der Möglichkeiten basiert auf den *Centroids* der Bündel (siehe Abschnitt 2.3) und stellt diese letztlich auf verschiedene Art und Weise dar.

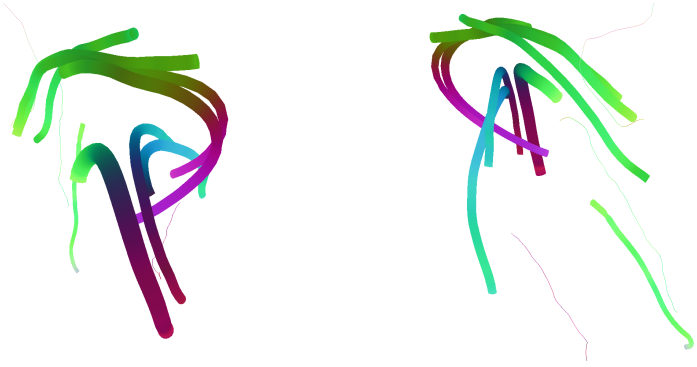
Liniendarstellung: Die Liniendarstellung (siehe Abbildung 4.2(b)) enthält minimale Informationen, d. h. lediglich über den räumlichen Verlauf des *Centroids* eines Bündels und ist analog zu der Visualisierung der Strömungslinien implementiert. Die Dicke der Linien wurde um den Faktor 3 angehoben, sodass eine Unterscheidung zu Strömungslinien möglich wird. Dieser Faktor wurde experimentell ermittelt und bietet einen guten Ausgleich zwischen Erkennbarkeit und Linienform, in Relation zu der Zylinderdarstellung. Diese Darstellungsart ist bevorzugt zu verwenden, wenn viele Bündel ermittelt wurden oder man sich einen Überblick über die tatsächlich exportierten Daten verschaffen möchte. Neben den dicken Linien sind zudem einige feinere zu sehen. Dies sind Strömungslinien, die als Rauschen klassifiziert wurden. Sie werden in diesem Beispiel unverändert dargestellt, können aber auch ausgeblendet werden.

Zylinderdarstellung: Eine andere Möglichkeit ist die Zylinderdarstellung (siehe Abbildung 4.2(c)). Diese entspricht einem *Centroid*, welcher auf einen bestimmten Radius ausgeweitet wurde. Dies bedeutet insbesondere, dass jedes Element einer Strömungslinie durch einen Kreis anstelle eines Punktes definiert wird, da dieser einem Schnitt durch den Zylinder entspricht.

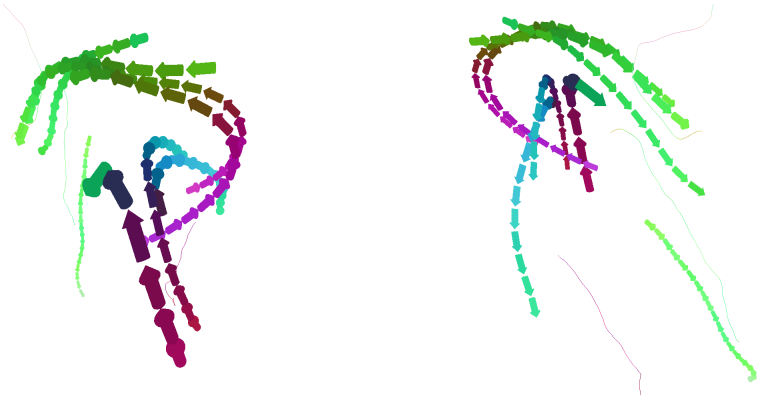


(a) Darstellung mittels Strömungslinien

(b) Bündel in der Liniendarstellung mit Rauschen



(c) Bündel in der Zylinderdarstellung mit Rauschen



(d) Bündel in der Pfeildarstellung mit Rauschen

Abbildung 4.2: Die verschiedenen Formen der Visualisierung.

Betrachtet man einen Kreis, dessen Mittelpunkt an einem Element v_i eines *Centroids* v innerhalb eines Vektorfeldes $\vec{f}$ angelegt ist, dann liegt dieser Kreis in einer Ebene, dessen Normalenvektor $\vec{n}$ definiert ist durch

$$\vec{n} = \frac{\vec{r}}{|\vec{r}|}$$

mit

$$\vec{r} = \frac{\vec{f}(v_i)}{|\vec{f}(v_i)|} + \frac{\vec{f}(v_{i+1})}{|\vec{f}(v_{i+1})|}.$$

Zwei aufeinanderfolgende Kreise werden miteinander verbunden, sodass aus je zwei Kreisen ein Zylinder mit nicht notwendigerweise parallelen Kappen entsteht, da die Normalenvektoren im Allgemeinen nicht linear abhängig sind. Da für je zwei aufeinanderfolgende Zylinder zwei dieser Kappen kongruent sind, können alle Zylinder entlang der Strömungslinie zu einem unterbrechungsfreien Schlauch zusammengefasst werden.

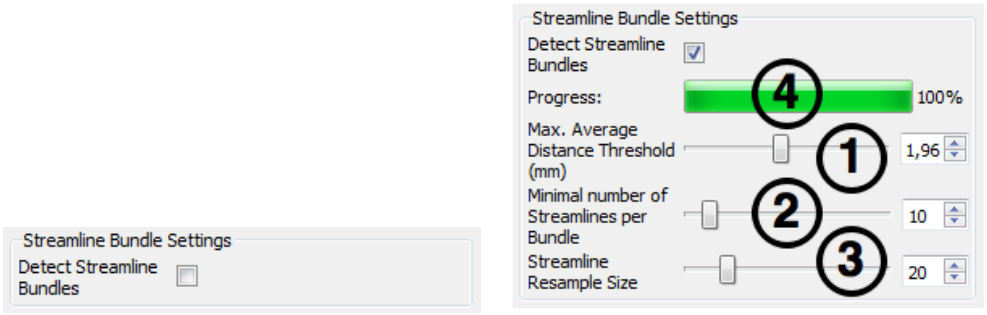
Pfeildarstellung: Eine weitere Möglichkeit ist die Pfeildarstellung (siehe Abbildung 4.2(d)). Diese erweitert in gewisser Form die Zylinderdarstellung, indem die einzelnen Zylinder verkürzt und mit Pfeilspitzen versehen werden, wodurch die Strömungsrichtung unmittelbar ersichtlich wird.

Bei der Zylinder- und Pfeildarstellung stößt man auf das Problem der Definition des Radius solcher geometrischer Körper. Bei den aus Zylindern bestehenden Schläuchen kann man sich zwei Szenarien überlegen. Entweder, ein solcher Schlauch hat einen einheitlichen Radius oder der Radius jedes Kreises entspricht den lokalen Ausmaßen der Strömung. Die Verwendung eines nicht konstanten Radius ist denkbar, allerdings ist dieser Ansatz anfällig gegenüber Strömungslinien, die nicht optimal dem Verlauf der zugehörigen Strömung folgen, was durch Rauschen im Datensatz verstärkt wird. In dieser Arbeit wird daher ein einheitlicher Radius verwendet, da so der Verlauf der Bündel gleichmäßig und übersichtlich visualisiert werden kann.

Zur Ermittlung des Radius wird eine grobe Approximation verwendet, was letztlich durch den Geschwindigkeitsvorteil begründet ist. Der Radius berechnet sich aus dem gemittelten Abstand des *Centroids* zu den enthaltenen Strömungslinien und wird ebenfalls verwendet, um die Länge der Pfeile in der Pfeildarstellung zu bestimmen. Dafür wird der *Centroid* so geresampled (siehe Abschnitt 3.2.2), dass die neue Segmentlänge gleich dem Radius multipliziert mit dem Faktor 4 ist. Die experimentelle Ermittlung dieses Faktors begründet sich durch den Wahrnehmungsvorteil in Bezug auf die Strömungsrichtung, den die Pfeildarstellung im Vergleich zur Zylinderdarstellung bieten soll. Es werden keine zusätzlichen Informationen dargestellt.

Nach dem Erstellen der Geometrie muss diese eingefärbt werden, um den räumlichen Verlauf wahrnehmen und eine Farbkodierung anwenden zu können.

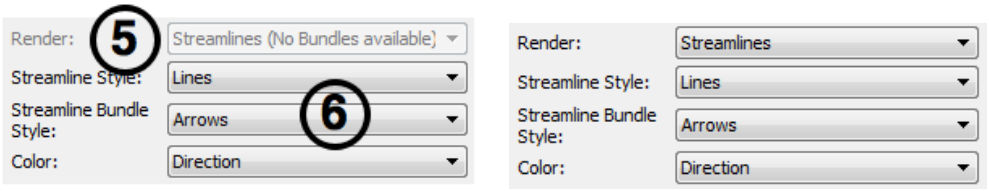
Die Informationen für die Farbkodierung der drei vorgestellten Möglichkeiten zur Visualisierung der Bündel werden dabei aus deren *Centroids* entnommen. Bei der Aufsummierung der enthaltenen Strömungslinien wird für jedes Element des *Centroids* die gemittelte lokale Geschwindigkeit bestimmt. Diese wird für das Einfärben der Bündel verwendet, analog zur Farbkodierung der Strömungslinien selbst (siehe Abschnitt 2.4.2). Bei der richtungsbasierten Farbkodierung wird dabei die Richtung der lokalen Strömung in einen Farbwert übersetzt. Der Anteil des normalisierten Richtungsvektors in x-Richtung kann z. B. als Rotanteil interpretiert werden. Die geschwindigkeitsbasierte Farbkodierung interpretiert dagegen die Magnitude des Richtungsvektors als Intensitätswert. Dieser kann dann, ausgehend vom maximalen Geschwindigkeitswert, in eine Farbe eines eindimensionalen Farbverlaufs übersetzt werden (siehe Abbildung 2.4).



(a) Gruppieren von Strömungslinien deaktiviert

(b) Gruppieren von Strömungslinien aktiviert

Abbildung 4.3: Die Gruppe von Einstellungsmöglichkeiten für die Bündelung im *StreamlineCreator*-Prozessor



(a) Visualisierung von Bündeln deaktiviert

(b) Visualisierung von Bündeln aktiviert

Abbildung 4.4: Die Einstellungsmöglichkeiten für die Visualisierung im *StreamlineRenderer3D*-Prozessor

4.2 Die Benutzerschnittstelle

Die Benutzerschnittstelle ist ein wichtiger Bestandteil, vor allem aus Sicht des Anwenders. Ziel ist es, eine möglichst intuitive Konfiguration des im vorangegangenen Abschnitt implementierten Algorithmus zu gewährleisten. Die Bedienung soll sich dabei ebenfalls an den bereits Vorhandenen Benutzerschnittstelle orientieren, um eine einheitliche Bedienung zu wahren.

Im Zuge der inkrementellen Implementierung ist das Gruppieren der Strömungslinien zu Bündeln optional. Die Funktionalität ist standardmäßig deaktiviert und kann in der Benutzerschnittstelle des StreamlineCreators aktiviert werden (siehe Abbildung 4.3(a)). Entscheidet sich der Benutzer, diese zu aktivieren, so wird eine neue Gruppe von Einstellungen sichtbar, welche ausschließlich für das Bündeln der Strömungslinien relevant sind (siehe Abbildung 4.3(b)). Der Parameter „Max. Avg. Distance Threshold“ (1) spezifiziert hierbei den Grenzwert in mm, der für die Zuordnung einer Strömungslinie zu einem Bündel ausschlaggebend ist. Der implementierte Algorithmus betrachtet alle Strömungslinien und versucht jede einem Bündel zuzuordnen. Dabei ist der durchschnittliche Abstand zwischen Bündel und Strömungslinie (siehe Abschnitt 3.2.1) der Wert, welcher den eingestellten Wert nicht überschreiten darf, um dem Bündel zugeordnet zu werden.

Der durchschnittliche Abstand kann die Abmessung des Datensatzes mit Gewissheit nicht überschreiten, wodurch der Parameter durch das Intervall $[0, d] \subset \mathbb{R}$ eingeschränkt werden kann, wobei d die Länge der Diagonalen durch den Datensatz darstellt. Auf diese Weise wird die Anzahl der möglichen Einstellung des Parameters, die zu einem unerwünschten Ergebnis führen können, unabhängig vom Datensatz stark reduziert.

Eine ähnliche Einschränkung existiert für den „Minimal number of Streamlines per Bundle“ Parameter (2). Dieser wird für die Entscheidung herangezogen, ob ein Bündel als solches weiterverarbeitet wird oder die enthaltenen Strömungslinien als Rauschen interpretiert werden. Enthält das Bündel nach dem vollständigen Durchlauf des Algorithmus mehr oder genauso viele Strömungslinien, wie mit dem Parameter eingestellt wurden, dann wird das Bündel als zentrale Strömung interpretiert und weiterverarbeitet. Enthält es weniger, so werden alle enthaltenen Strömungslinien als Rauschen im Datensatz interpretiert und das Bündel wird verworfen. Der Wertebereich ist hierbei eingeschränkt durch: $[0, n] \subset \mathbb{R}$, wobei n die Anzahl an Strömungslinien repräsentiert.

Der Parameter „Streamline Resample Size“ (3) spezifiziert die Genauigkeit der Diskretisierung der Strömungslinien. Die zuvor gefundenen Strömungslinien werden im Normalfall durch eine unterschiedliche Anzahl von Elementen und verschieden langen Abständen zwischen den Elementen beschrieben. Der zum Gruppieren verwendete Algorithmus sieht dagegen für alle Strömungslinien eine identische Anzahl an Elementen jeder Strömungslinie und identisch lange Abstände zwischen den Elementen innerhalb einer Strömungslinie vor, um zu

funktionieren. Daher ist es notwendig, alle Strömungslinien vor der Gruppierung umzurechnen, um diese Bedingungen zu erfüllen. Der Parameter gibt dabei an, auf wie viele Elemente jede Strömungslinie umgerechnet wird. Der Wertebereich ist hierbei eingeschränkt durch $[2, 100] \subset \mathbb{N}$. Die untere Schranke wurde auf den Wert zwei beschränkt, damit Strömungslinien bei der Umrechnung nicht degenerieren können. Die obere Schranke ist auf den Wert 100 gesetzt worden, da sich experimentell gezeigt hat, dass eine weitere Erhöhung nicht zu einer Verbesserung des Ergebnisses führt.

Weiterhin gibt es einen Fortschrittsbalken (4), der einen Indikator für die Berechnungsdauer des Algorithmus darstellt. Dieser kann für den Nutzer hilfreich sein, da die Berechnung in Abhängigkeit von der maximalen Anzahl an Strömungslinien, den Parametern (1) und (3), sowie der Leistung des ausführenden Systems, einige Zeit in Anspruch nehmen kann. So kann besser eingeschätzt werden, ob eventuell ungünstige Konfigurationen der oben genannten Parameter vorgenommen wurden. Parameter (1) ist dabei ausschlaggebend, da sich bei der Einstellung 0.00, die Laufzeitkomplexität stark verschlechtert (siehe Abschnitt 3.2.3).

Startet man daraufhin die Berechnung der Strömungslinien mit der erstellten Konfiguration, dann sind zunächst keine Bündel vorhanden, die visualisiert werden können. Aus diesem Grund wird im *StreamlineRenderer3D*-Prozessor (siehe Abbildung 4.4) die Option (5) deaktiviert. Dadurch wird implizit die Strömungsliniendarstellung gewählt. Sollte die Berechnung allerdings abgeschlossen und Bündel gefunden worden sein, so wird die Option aktiviert und der Benutzer kann zwischen den, im vorangegangenen Abschnitt erklärten, Darstellungsmöglichkeiten wählen (6).

5 Fazit und Ausblick

In dieser Arbeit wurde der QuickBundles-Algorithmus implementiert und in das Framework Voreen integriert. Dieser kann die zuvor berechneten Strömungslinien in sehr kurzer Zeit in Bündel überführen und somit Ströme sichtbar machen, die zuvor aufgrund der großen Anzahl an ungeordneten Strömungslinien nicht sichtbar wären. Geht man von einem geeigneten Datensatz aus, erzielt der Algorithmus überzeugende Ergebnisse.

Diese werden in anschaulicher Art und Weise durch verschiedene Darstellungsmöglichkeiten der *Centroids* visualisiert. Dabei kann zur Zeit zwischen Linien-, Zylinder- oder Pfeildarstellung gewählt werden, die jede für sich situationsabhängig einen Wahrnehmungsvorteil im Vergleich zu der puren Strömungsliniendarstellung bieten kann. Die eingangs erwähnte Eignung des Datensatzes ist jedoch ein limitierender Faktor für das Ergebnis des Algorithmus. Dessen Anwendung ist nicht sinnvoll, wenn der zugrundeliegende Datensatz aus vielen, kurzen Strömungslinien besteht. Ist deren Länge jedoch möglichst ausgedehnt, d. h. diese durchziehen beispielsweise ein Gefäß der Länge nach, dann ist ein zufriedenstellendes Ergebnis sehr wahrscheinlich. Hier kommt zum Tragen, dass der Algorithmus mit sehr wenigen Parametern auskommt. Von diesen ist im Wesentlichen nur der Schwellwert für die zugrundeliegende MDF-Metrik für das Ergebnis von signifikanter Bedeutung. Der Anwender muss jedoch abhängig vom Datensatz und seinen Erwartungen entscheiden, ob er einen hohen oder einen niedrigen Schwellwert wählt. Die erste Option führt dabei zu wenigen Bündeln aber kann zu einem größeren Fehler in der Übereinstimmung zu tatsächlich vorhandenen Strömungen führen. Letztere reduziert dagegen den Fehler, führt aber zu mehreren Bündeln.

An diesem Punkt lässt sich Potential für Automatisierung und Verbesserung festhalten. So lassen sich sinnvolle Voreinstellungen der Parameter für einen Datensatz bestimmen. Für die „Streamline Resample Size“ lässt sich feststellen, wie gekrümmt die Strömungslinien im Datensatz sind, d. h. wie groß der Winkel zwischen zwei Richtungsvektoren aufeinanderfolgender Elemente durchschnittlich oder maximal ist. Der Schwellwert für das Kategorisieren eines Bündels als Rauschen könnte durch eine statistische Auswertung aller gefundenen Bündel ermittelt werden. Diese Automatisierungen erfolgen jedoch auf Kosten der Laufzeit, wodurch deren Umsetzung zur Zeit nicht sinnvoll erscheint. Durch die technologische Entwicklung oder die Parallelisierung der Prozesse könnte dies jedoch eine ernstzunehmende Verbesserungsmöglichkeit darstellen.

Das automatisierte Bestimmen des bereits erwähnten Schwellwerts für die

MDF-Metrik erfordert jedoch eine intelligente Analyse des kompletten Datensatzes, womit die Anwendung des QuickBundles-Algorithmus hinfällig werden würde. Die Einstellung dieses Wertes bleibt somit dem Anwender überlassen.

Eine weitere Verbesserungsmöglichkeit findet sich in der Bestimmung des Radius in der Zylinder- und Pfeildarstellung. Dieser wird momentan durch eine sehr grobe Approximation ermittelt. Hier sollte von einem einheitlichen Radius Abstand genommen und algorithmisch eine konvexe Hülle über die Strömungslinien eines Bündels berechnet werden, sodass die räumliche Ausdehnung einer Strömung exakt visualisiert werden kann. Die Länge der Pfeile sollte von der Krümmung der Strömungslinien abhängen, sodass in stark gekrümmten Regionen der durch das Überspringen von Elementen entstehende Fehler geringer ausfällt. Zudem würde beispielsweise ein Phong-Shading [6] die räumliche Wahrnehmung bei der Pfeildarstellung stark verbessern.

Literaturverzeichnis

[1] Anderson, Jeff R, Orlando Diaz, Richard Klucznik, Y Jonathan Zhang, Gavin W Britz, Robert G Grossman, Nan Lv, Qinghai Huang und Christof Karmonik: *Validation of computational fluid dynamics methods with anatomically exact, 3D printed MRI phantoms and 4D pcMRI.* In: *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Seiten 6699–6701. IEEE, 2014.

[2] Bovenkamp, Philipp Rene, Tobias Brix, Florian Lindemann, Richard Holtmeier, Desiree Abdurrachim, Michael T Kuhlmann, Gustav J Strijkers, Jörg Stypmann, Klaus H Hinrichs und Verena Hoerr: *Velocity mapping of the aortic flow at 9.4 T in healthy mice and mice with induced heart failure using time-resolved three-dimensional phase-contrast MRI (4D PC MRI).* Magnetic Resonance Materials in Physics, Biology and Medicine, 28(4):315–327, 2015.

[3] Garyfallidis, Eleftherios, Matthew Brett, Marta Morgado Correia, Guy B Williams und Ian Nimmo-Smith: *Quickbundles, a method for tractography simplification.* Frontiers in neuroscience, 6:175, 2012.

[4] Jain, Anil K: *Data clustering: 50 years beyond K-means.* Pattern recognition letters, 31(8):651–666, 2010.

[5] Meyer-Spradow, Jennis, Timo Ropinski, Jörg Mensmann und Klaus Hinrichs: *Voreen: A rapid-prototyping environment for ray-casting-based volume visualizations.* IEEE Computer Graphics and Applications, 29(6):6–13, 2009.

[6] Phong, Bui Tuong: *Illumination for computer generated pictures.* Communications of the ACM, 18(6):311–317, 1975.

[7] Spurk, Joseph H und Nuri Aksel: *Strömungslehre: Einführung in die Theorie der Strömungen.* Springer-Verlag, 8 Auflage, 2010.

[8] Udwadia, Firdaus E und Artin Farahani: *Accelerated Runge-Kutta Methods.* Discrete Dynamics in Nature and Society, 2008, 2008.

[9] Woo, Mason, Jackie Neider, Tom Davis und Dave Shreiner: *OpenGL programming guide: the official guide to learning OpenGL, version 1.2.* Addison-Wesley Longman Publishing Co., Inc., 1999.

Eidesstattliche Erklärung

Hiermit versichere ich, dass die vorstehende Arbeit selbstständig verfasst worden ist, dass keine anderen Quellen und Hilfsmittel als die angegebenen benutzt worden sind und dass die Stellen der Arbeit, die anderen Werken – auch elektronischen Medien – dem Wortlaut oder Sinn nach entnommen wurden, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht worden sind.

(Datum, Unterschrift)

Ich erkläre mich mit einem Abgleich der Arbeit mit anderen Texten zwecks Auffindung von Übereinstimmungen sowie mit einer zu diesem Zweck vorzunehmenden Speicherung der Arbeit in eine Datenbank einverstanden.

(Datum, Unterschrift)